Scott Jackson, PNNL
David Jackson, Ames Lab
Brett Bode, Ames Lab

Scalable Systems Software Node Object Specification

Status of this Memo

This is a specification of the node object to be used by Scalable Systems Software compliant components. It is envisioned for this specification to be used in conjunction with the SSSRMAP protocol with the node object passed in the Data field of Requests and Responses. Queries can be issued to a node-cognizant component in the form of modified XPATH expressions to the Get field to extract specific information from the node object as described in the SSSRMAP protocol.

Abstract

This document describes the syntax and structure of the SSS node object. This node model takes into account various node property categories such as whether it represents a configured, available or utilized property.

Table of Contents

1. Introduction

This specification proposes a standard XML representation for a node object for use by the various components in the SSS Resource Management System. This object will be used in multiple contexts and by multiple components. It is anticipated that this object will be passed via the Data Element of SSSRMAP Requests and Responses.

1.1 Goals

There are several goals motivating the design of this representation.

It needs to be inherently flexible. We recognize we will not be able to exhaustively include the ever-changing node properties and capabilities that constantly arise.

The same node object should be used at all stages of its lifecycle. This object needs to distinguish between configured, available and utilized properties of a node.

Its design takes into account the properties and structure required to function in a meta or grid environment. It should eventually include the capability of resolving namespace and locality issues, though the earliest versions will ignore this requirement.

One should not have to make multiple queries to obtain a single piece of information -- i.e. there should not be two mutually exclusive ways to represent a node resource.

Needs to support resource metric as well as unit specifications.

1.2 Examples

1.2.1 Simple Example

This example shows a simple expression of the Node object.

```
<Node>
        <NodeId>Node64</NodeId>
```

```
        <Configured>
                <Processors>2</Processors>
                <Memory>512</Memory>
        </Configured>
</Node>
```

1.2.2 Elaborate Example

This example shows a more elaborate Node object.

```
<Node>
        <NodeId>64</NodeId>
        <Name>Netpipe2</Name>
        <Feature>BigMem</Feature>
        <Feature>NetOC12</Feature>
        <Opsys>AIX</Opsys>
        <Arch>Power4</Arch>
        <Configured>
                <Processors>16</Processors>
                <Memory units=”MB”>512</Memory>
                <Swap>512</Swap>
        </Configured>
        <Available>
                <Processors>7</Processors>
                <Memory metric=”Instantaneous”>143</Memory>
        </Available>
        <Utilized>
                <Processors wallDuration=”3576”>8</Processors>
                <Memory metric=”Average” wallDuration=”3576”>400</Memory>
        </Utilized>
</Node>
```

2. Conventions used in this document

2.1 Keywords

The keywords “MUST”, “MUST NOT”, “REQUIIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in RFC2119.

2.2 Table Column Interpretations

In the property tables, the columns are interpreted to have the following meanings:

Element Name:	Name of the XML element (xsd:element)
Type:	Data type defined by xsd (XML Schema Definition) as:
String	xsd:string (a finite length sequence of printable characters)
Integer	xsd:integer (a signed finite length sequence of decimal digits)
Float	xsd:float (single-precision 32-bit floating point)
Boolean	xsd:boolean (consists of the literals “true” or “false”)
DateTime	xsd:dateTime (discreet time values are represented in ISO 8601 extended format CCYY-MM-DDThh:mm:ss where "CC" represents the century, "YY" the year, "MM" the month and "DD" the day. The letter "T" is the date/time separator and "hh", "mm", "ss" represent hour, minute and second respectively. This representation may be immediately followed by a "Z" to indicate Coordinated Universal Time (UTC) or, to indicate the time zone, i.e. the difference between the local time and Coordinated Universal Time, immediately followed by a sign, + or -, followed by the difference from UTC.)
Duration	xsd:duration (a duration of time is represented in ISO 8601 extended format PnYnMnDTnHnMnS, where nY represents the number of years, nM the number of months, nD the number of days, 'T' is the date/time separator, nH the number of hours, nM the number of minutes and nS the number of seconds. The number of seconds can include decimal digits to arbitrary precision.)
Description:	Brief description of the meaning of the property
Appearance:	This column indicates whether the given property has to appear within the parent element. It assumes the following meanings:
MUST	This property is REQUIRED when the parent is specified.
SHOULD	This property is RECOMMENDED when the parent is specified.
MAY	This property is OPTIONAL when the parent is specified.
Compliance:	The column indicates whether a compliant implementation has to support the given property.
MUST	A compliant implementation MUST support this property.
SHOULD	A compliant implementation SHOULD support this property.
MAY	A compliant implementation MAY support this property.

Categories:	Some properties may be categorized into one of several categories. Letters in this column indicate that the given property can be classified in the the following property categories.
C	This property can be encompassed in a Configured element.
A	This property can be encompassed in an Available element.
U	This property can be encompassed in a Utilized element.

2.3 Element Syntax Cardinality

The cardinality of elements in the element syntax sections may make use of regular expression wildcards with the following meanings:

*	Zero or more occurrences
+	One or more occurrences
?	Zero or one occurrences

The absence of one of these symbols implies one and only one occurrence.

3. The Node Model

The primary element within the node model is a node. One can speak of some node properties as being a configured, available or utilized property of the node.

4. Node Element

The Node element is the root element of a node object and is used to encapsulate a node.

- A node object MUST have exactly one Node element.
- A compliant implementation MUST support this element.
- A node MUST specify one or more Node Properties.

4.1 Uncategorized Node Properties

Uncategorized Node Properties are properties that apply to the node as a whole and do not need to be distinguished between being configured, available or utilized. These include the node id and other optional node properties.

4.1.1 Simple Node Properties

Simple (unstructured) node properties are enumerated in Table 1.

Table 1 Simple Node Properties

Element Name	Type	Description	Appearance	Compliance
NodeId	String	Node identifier	MUST	MUST
Name	String	Node name or pattern.	MAY	MAY
OpSys	String	Operating System	MAY	SHOULD
Arch	String	Architecture	MAY	SHOULD
Description	String	Description of the node	MAY	MAY
State	String	State of the node. Valid states may include "Offline", "Configured", "Unknown", "Idle", "Busy".	SHOULD	MUST
Features	String	Arbitrary named features of the node (comma-delimited string).	MAY	SHOULD

4.1.2 Extension Element

The Extension element provides a means to pass extensible properties with the node object. Some applications may find it easier to deal with a named extension property than discover and handle elements for which they do not understand or anticipate by name.

- A compliant implementation MAY support this element.
- This element MUST have a name attribute that is of type String and represents the name of the extension property. A compliant implementation MUST support this attribute if this element is supported.
- This element MAY have a type attribute that is of type String and provides a hint about the context within which the property should be understood. A compliant implementation SHOULD support this attribute if this element is supported.
- The character content of this element is of type String and is the value of the extension property.

The following is an example of an Extension element:

<Extension type="Chemistry" name="Software">NWChem</Extension>

4.2 Property Categories

Certain node properties (particularly consumable resources) need to be classified as being in a particular category. This is done when it is necessary to distinguish between a property that is configured versus a property that is available or utilized. For example, a node might be configured with 16 processors. At a particular time, 8 might be utilized, 7 might be available and 1 disabled. When a node property must be categorized to be understood properly, the property MUST be enveloped within the appropriate Property Category Element.

4.2.1 Configured Element

A configured node property reflects resources pertaining to the node that could in principle be used though they may not be available at this time. This information could be used to determine if a job could ever conceivably run on a given node.

- A compliant implementation MUST support this element.

The following is an example of using Configured Properties:

```
<Configured>
    <Processors>16</Processors>
    <Memory units=”MB”>512</Memory>
</Configured>
```

4.2.2 Available Element

An available node property refers to a resource that is currently available for use.

- A compliant implementation SHOULD support this element.

The following is an example of specifying available properties:

```
<Available>
    <Processors>7</Processors>
    <Memory units=”MB”>256</Memory>
</Available>
```

4.2.3 Utilized Element

A utilized node property reflects resources that are currently utilized.

- A compliant implementation SHOULD support this element.

The following is an example of specifying utilized properties:

```
<Utilized>
    <Processors>8</Processors>
    <Memory metric=”Average”>207</Memory>
</Utilized>
```

4.3 Categorized Node Properties

4.3.1 Consumable Resources

Consumable Resources are a special group of node properties that can have additional attributes and can be used in multiple categories. In general a consumable resource is a resource that can be consumed in a measurable quantity.

- A consumable resource MUST be categorized as being a configured, available or utilized node property by being a child element of a Configured, Available or Utilized element respectively.
- A consumable resource MAY have a units attribute that is of type String that specifies the units by which it is being measured. If this attribute is omitted, a default unit is implied. A compliant implementation MAY support this attribute if the element is supported.
- A consumable resource MAY have a metric attribute that is of type String that specifies the type of measurement being described. For example, the measurement may be a Total, an Average, a Min or a Max. A compliant implementation MAY support this attribute if the element is supported.
- A consumable resource MAY have a wallDuration attribute of type Duration that indicates the amount of time for which that resource was used. This need only be specified if the resource was used for a different amount of time than the wallDuration for the step. A compliant implementation MAY support this attribute if the element is supported.
- A consumable resource MAY have a consumptionRate attribute of type Float that indicates the average percentage that a resource was used over its wallDuration. For example, an overbooked SMP running 100 jobs across 32 processors may wish to scale the usage and charge by the average fraction of processor usage actually delivered. A compliant implementation MAY support this attribute if the element is supported.

A list of simple consumable resources is listed in Table 2.

Table 2 Consumable Resource Node Properties

Element Name	Type	Description	Appearance	Compliance	Categories
Processors	Integer	Number of processors.	MAY	MUST	CAU
Memory	Float	Amount of memory.	MAY	SHOULD	CAU
Disk	Float	Amount of disk.	MAY	SHOULD	CAU
Swap	Float	Amount of virtual memory.	MAY	MAY	CAU
Network	Float	Amount of network.	MAY	MAY	CAU

The following are two examples for specifying a consumable resource:

<Memory metric=”Max” units=”GB”>483</Memory>

<Processors wallDuration=”1234” consumptionRate=”0.63”>4</Processors>

4.3.2 Resource Element

In addition to the consumable resources enumerated in the above table, an extensible consumable resource is defined by the Resource element.

- A compliant implementation SHOULD support this element.
- This element MAY appear zero or more times within a given set of node properties.
- Like the other consumable resources, this property MUST be categorized as a configured, available or utilized property by being encompassed in the appropriate elements.
- This element is of type Float.
- It shares the other same properties and attributes as the other consumable resources but it requires an additional name (and optional type) attribute to describe it.
- This element MUST have a name attribute of type String that indicates the type of consumable resource being measured. A compliant implementation MUST support this attribute if the element is supported.
- This element MAY have a type attribute of type String that distinguishes it within a general resource class. A compliant implementation SHOULD support this attribute if the element is supported.

The following are two examples for specifying a Resource element:

<Resource name=”License” type=”MATLAB”>1</Resource>

<Resource name=”Telescope” type=”Zoom2000” wallDuration=”750” metric=”KX”>10</Resource>

Appendix A

Units of Measure Abbreviations

Abbreviation	Definition	Quantity
B	byte	1 byte
KB	Kilobyte	2^10 bytes
MB	Megabyte	2^20 bytes
GB	Gigabyte	2^30 bytes
TB	Terabyte	2^40 bytes
PB	Petabyte	2^50 bytes
EB	Exabyte	2^60 bytes
ZB	Zettabyte	2^70 bytes
YB	Yottabyte	2^80 bytes
NB	Nonabyte	2^90 bytes
DB	Doggabyte	2^100 bytes